Name: Maura Kieft
ID: 103 947 905

CSCI 3104 **Profs. Grochow & Layer**
Problem Set 7 **Spring 2019, CU-Boulder**

Describe and analyze an algorithm (explain how it works, give pseudocode if necessary, derive its running time and space usage, and prove its correctness) that takes $O(V + E)$ time and space to convert G into G', and thereby will solve any of the parrot's questions. Assume both G and G' are stored as adjacency lists.

Hermione's hints: Don't assume adjacencies `Adj[u]` are ordered in any particular way, and remember that you can add edges to the list and then remove ones you don't need.

An algorithm to replace every directed multiedge by a single edge and to remove all self loops from an adjacency list can start by looping through the adjacency list for each node and marking each element in the respective node list as visited. If the edge in the list hasn't been visited, but is already represented in the list for the node, it is deleted, and if that edge is less than the previous one it is swapped to be put in the correct order and, thus, sorting the list from lower edge value to the higher edge value. While also doing this check, we check to make sure the edge is not equal to the node which will remove the self loops from the graph.

Since we loop through the entire adjacency list to perform the checks and detetions of edges, this algorithm should run in $O(V+E)$ time.

Using the examples provided in the question, we can prove this algorithms correctness.
The adjacency list for the graph G1 is

1	3 2
2	1 3 3 3
3	4
4	4 5 2 5 1
5	3 1

next page →